Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Complete local search with limited memory algorithm for no-wait job shops to minimize makespan

Jie Zhu *, Xiaoping Li, Qian Wang

School of Computer Science and Engineering, Southeast University, 210096 Nanjing, PR China
Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, 210096 Nanjing, PR China

ARTICLE INFO

Article history:
Received 19 October 2007
Accepted 16 September 2008
Available online 23 September 2008

Keywords:
No-wait
Job shop
Timetabling
Sequencing

ABSTRACT

In this paper, no-wait job shop problems with makespan minimization are considered. It is well known that these problems are strongly NP-hard. The problem is decomposed into the sequencing and the timetabling components. Shift timetabling is developed for the timetabling component. An effective method, CLLM (complete local search with limited memory), is presented by integrating with shift timetabling for the sequencing component. Experimental results show that CLLM outperforms all the existing effective algorithms for the considered problem with a little more computation time.

1. Introduction

In this paper, the no-wait job shop with makespan minimization is considered, in which consecutive operations of each job must be performed continuously without any interruption. No-wait constraints usually arise from requirements for processing environments or characteristics of jobs. Typical examples are iron being immediately stanched while hot in metallurgical processes and unstable intermediate products or absence of intermediate storage capacity in chemical processes. Such job shop problems widely exist in real-life applications, such as chemical and pharmaceutical industries [18,19], steel production [27], computer systems [20], semiconductor testing facilities [17], surgical cases [4], and building industries [9].

The traditional job shop problem, in which there is no no-wait constraint, is NP-hard in the strong sense [13]. Detailed surveys have been conducted in [3,11] for the results on the problem in the past decades, but little attention has been paid to the problem considered in this paper. The review given by Hall and Sriskandarajah [10] showed that the considered problem is difficult, especially for large-size instances. It was proven to be NP-hard in the strong sense even for two-machine cases [21]. Bansal et al. [2] showed that, if each job has at most two operations and the number of machines is a constant, the problem with the makespan objective function admits a polynomial time approximation scheme. For the job shop scheduling problem with blocking and/or no-wait constraints, Meloni et al. [16] developed a rollout method in terms of a general alternative graph model based on a look-ahead strategy. Additionally, some approximate algorithms (tabu search [14], simulated annealing [18] and rolling horizon procedures [17]) have been developed for specific production systems, such as the galvanic industry, production of pharmaceutics and semiconductor testing facilities.

Recently, several algorithms have been presented for no-wait job shops with makespan minimization. Besides the branch and bound algorithm proposed by Mascis and Pacciarelli [15], it seems that VNS (variable neighborhood search) [22], GASA (hybrid simulated annealing/genetic algorithm) [22], CLM (complete local search with memory) [6] and tabu search [24] are the most effective or efficient ones. Among these effective approaches, the problem is usually decomposed into two sub-problems: the timetabling problem and the sequencing problem. VNS depends on special structures of problems, and its speed is very fast. GASA is a non-deterministic hybrid algorithm, which is more time-consuming but also more effective than VNS. Both VNS and GASA adopt non-delay timetabling for the timetabling problem. For the sequencing problem, VNS searches the optimal results iteratively by a *k*-insertion neighborhood until no better solution can be found. GASA integrates genetic algorithm (GA) with simulated annealing (SA) for the sequencing problem. CLM also takes advantage of non-delay timetabling. Furthermore, enhanced timetabling is introduced in CLM. This is when non-delay timetabling is performed on "reversed jobs" (the processing route of every job is reversed). For the sequencing problem, CLM generates a neighborhood by 1-insertion, and all the exploited solutions are

* Corresponding author. Address: School of Computer Science and Engineering, Southeast University, 210096 Nanjing, PR China.
E-mail addresses: zhujie_hades@163.com (J. Zhu), xpli@seu.edu.cn (X. Li), qwang@seu.edu.cn (Q. Wang).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.09.015

recorded to avoid re-calculating. CLM seems to be the best algorithm so far for the considered problem.

Based on CLM, CLLM is presented in this paper. In contrast to CLM, VNS and GASA, which treat the two sub-problems separately, CLLM solves them integrally. Shift timetabling is proposed to handle the timetabling problem. Then, shift timetabling is integrated with a complete local search to solve the sequencing problem.

The rest of this paper is organized as follows. The considered problem is described in Section 2. Section 3 constructs shift timetabling. Combined with the shift timetabling approach, a new local search approach is proposed in Section 4. In Section 5, the experimental results are shown, followed by conclusions in Section 6.

2. Problem description

A no-wait job shop is a scheduling problem in which n jobs are processed on m machines and each job has a specific processing route. Each job has to be processed on each machine exactly once. *Processing sequence* is a permutation π of the set of jobs, meaning that jobs should be arranged orderly by their index in π. A processing sequence is denoted as a vector $(\pi_{[1]},\dots,\pi_{[n]})$. A processing of a job on a machine is called *operation*. Besides, some notations to be used in this paper are given.

$n \in N$	number of jobs
$m \in N$	number of machines
$J = \{J_1, J_2, \dots, J_n\}$	set of jobs
$M = \{M_1, M_2, \dots, M_m\}$	set of machines
$\pi_{[i]} \in \{J_1, J_2, \dots, J_n\}$	the ith job in π
$m_{i,k} \in \{M_1, M_2, \dots, M_m\}$	machine on which the kth operation of J_i is processed
$p_{i,k} \in N$	processing time of the kth operation of J_i
$t_{\pi_{[i]}} \in N$	start time of $\pi_{[i]}$
$P_{i,j} = \sum_{k=1}^{j} p_{i,k}$	cumulated processing time of J_i (including the jth operation)
$T_\pi = (t_{\pi_{[1]}},\dots,t_{\pi_{[n]}})$	timetable of π
$t_i^k \in N$	start time of J_i on M_k
$p_i^k \in N$	processing time of J_i on M_k
$E_{i,j} = \{\{k,l\} \mid m_{i,k} = m_{j,l}\}$	pairs of operations of jobs J_i and J_j to be processed on the same machine

Furthermore, the following constraints should hold according to the no wait and job shop properties.

- *Sequence*: Each job must be processed in order of its operations, and no interruption (preemption) of an operation is allowed.
- *Synchronicity*: No job may be processed by two machines at the same time, and no machine may process two jobs at the same time.
- *No-wait*: There must be no waiting time between two consecutive operations of the same job.

Therefore, the timetable of the considered problem is based on start times of the first operations. Mapping a set of start times to the given processing sequence is called a *feasible schedule*, if the aforementioned constraints are met. A feasible schedule is denoted as $S_\pi = \langle \pi, T_\pi \rangle$. An *optimal schedule* is a feasible one that minimizes the makespan.

For the considered problem, which is NP-hard in the strong sense, effective algorithms usually decompose it into two sub-problems: (1) the *sequencing problem*, in which a processing sequence of an optimal schedule is found for a given no-wait job shop problem, and (2) the *timetabling problem*, in which a feasible set of start times of the jobs is found to minimize makespan for the processing sequence obtained from (1). Both sub-problems have been proven to be NP-hard in the strong sense [23]. Sriskandarajah and Ladet [25] investigated the computational complexity of no-wait shops scheduling problems and concluded that finding optimal schedules for no-wait job shops is NP-hard even in two-machine no-wait job shops.

3. Timetabling methods

Because it is commonly believed that the sequencing problem is much more difficult than the timetabling problem, more attention has been paid to the sequencing problem. Existing algorithms spend the bulk of the computation time on the sequencing problem while employing simple strategies for the timetabling problem. In fact, effectiveness of the no-wait job shop also depends greatly on timetabling methods. In this paper, shift timetabling is introduced based on non-delay timetabling and enhanced timetabling.

3.1. Non-delay timetabling

Schuster [24] defined the non-delay timetabling formally as follows:

Definition 1. Non-delay timetablingGiven a timetabling problem with processing sequence $(\pi_{[1]},\dots,\pi_{[n]})$, set $t_{\pi_{[1]}} \leftarrow 0$ and compute successively for $i = 1,\dots,n$ the solutions of

$$\min \quad t_{\pi_{[i]}} \tag{1a}$$

$$\text{s.t.} \quad t_{\pi_{[i]}} \geqslant t_{\pi_{[i-1]}}, \tag{1b}$$

$$t_{\pi_{[j]}} - t_{\pi_{[i]}} \geqslant P_{\pi_{[i]},k} - P_{\pi_{[j]},l-1} \quad \text{or} \quad t_{\pi_{[i]}} - t_{\pi_{[j]}} \geqslant P_{\pi_{[j]},l} - P_{\pi_{[i]},k-1}$$
$$\text{for all } (k,l) \in E_{\pi_{[i]},\pi_{[j]}}, \tag{1c}$$

$$j < i, \quad j \in \{1,\dots,n\}. \tag{1d}$$

The constraint (1b) is called *ordinal constraint*. Conditions (1c) prohibit a machine to process two operations at the same time, i.e. no conflict (overlapping on the Gantt chart) exists. Obviously, the makespan obtained by non-delay timetabling is just an upper bound of the sub-problem, which can be further improved.

3.2. Enhanced timetabling

Enhanced timetabling explores the symmetry of the considered problem to improve non-delay timetabling. It performs on inverse instances and inverse sequences as well as the original ones. *Inverse instance* is the instance identical to the original one but each job is processed in the reversed route.

Example 1. For a three-job, five-machine problem, the original processing route matrix and its corresponding route matrix of the inverse instance are shown below

$$[m_{i,k}]_{3\times5} = \begin{bmatrix} 3 & 1 & 2 & 4 & 0 \\ 4 & 3 & 1 & 2 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}, \quad [\hat{m}_{i,k}]_{3\times5} = \begin{bmatrix} 0 & 4 & 2 & 1 & 3 \\ 0 & 2 & 1 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

$(\pi_{[n]},\dots,\pi_{[1]})$ is the *inverse processing sequence*of processing sequence $(\pi_{[1]},\dots,\pi_{[n]})$.

Enhanced timetabling includes three phases. (1) Non-delay timetabling is performed on the original instance. (2) Non-delay timetabling is conducted on the inverse instance for the inverse processing sequence. (3) The better of the two trials is chosen as the final solution. Obviously, non-delay timetabling is included in enhanced timetabling. Therefore, enhanced timetabling dominates non-delay timetabling.

Though start times can be easily calculated by enhanced timetabling or non-delay timetabling, many gaps may be formed under the ordinal constraint, which may be filled with some job(s).